Draft

Explaining Prediction Models and Individual Predictions with Feature Contributions

Erik Štrumbelj,Igor Kononenko

1 Introduction

Classification and regression prediction models are an important component of decision support systems. Such models are often complex and non-transparent to the user and require additional post-processing. The purpose of such post-processing is to obtain a better understanding of the model and, in practical applications, increase the end-user's level of trust in the model. The latter is especially important in risk-sensitive domains such as medicine, where experts are reluctant in trusting prediction model's predictions without additional explanation.

Most explanation methods are model specific. Some models, such as decision and regression trees, rules, and nearest neighbors-based methods, are self-explanatory, provided that the number of nodes/rules/dimensions is not too large. For large trees and/or sets of rules, additional post-processing methods have been developed that help reduce the models size while minimizing the loss of informativeness [1, 2, 3, 4, 5, 6]. Linear regression and other additive models can be additionally explained by plotting the marginal effect of each input variable. Because of additivity, the actual prediction is the sum of individual marginal effects, which also makes such a visualization a tool for graphical computation of predictions - a nomogram. Additivity has been exploited to provide an explanation for the Naive Bayes classifier [7, 8, 9], linear SVM [10], logistic regression [11], Cox regression models [12], and additive models in general [13]. Complex models, such as artificial neural networks [14, 15, 16, 17, 18] and SVM (Support Vector Machines) [19, 20, 10, 21, 22, 23, 24, 25] received most attention, because they are often very successful but non-transparent.

In this paper we focus on a general approach for explaining prediction models. General in the sense that it can be applied to any type of prediction models. Such approaches must essentially treat the model as a black box, restricted to explaining the model only through changing the inputs and observing the changes in the output. While such approaches can not exploit the specifics of the model, they have the advantage of being applicable to any type of model and providing uniform explanations. This facilitates comparison of different types of models and, in practical applications, eliminates the need to replace the explanation method when the underlying model is changed or replaced.

The key component of general explanations are the contributions of individual input features. The prediction for a particular instance is additionally explained by assigning to each feature a number which denotes its share in the prediction. For each feature, such contributions can be aggregated to plot the feature's average contribution against the feature's value. Such a plot provides an overview of the model and is similar to plotting the marginal effect for an additive model.

To describe our problem setting and provide a better understanding of the idea of feature contributions, we start with a simple illustrative example - the linear regression model:

$$f(x) = f(x_1, ..., x_n) \approx y = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n$$

If the input features are standardized, the coefficient β_i can be interpreted as the $i-$th feature's global importance. However, the coefficient itself does not provide much in terms of explaining the model. In terms of explaining a prediction or the model, we are more interested in how a particular value influences the prediction. We turn to the following expression

$$\varphi_i(x) = \beta_i x_i - \beta_i E[X_i], \tag{1}$$

which is known as the situational importance of $X_i = x_i$ [26].

The situational importance is the difference between what a feature contributes when its value is x_i and what it is expected to contribute. If the situational importance is positive, then the feature has a positive contribution (increases the prediction for this particular instance), if it is negative, then the feature has a negative contribution (decreases the prediction), if it is 0, it has no contribution.

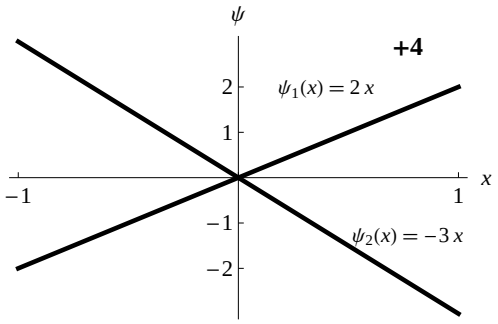


Figure 1: The marginal effects of the two input features for the model $f(x_1, x_2) = 2x_1 - 3x_2 + 4$. Both input features' functions are shown on the same graph. Such plots can be used for semi-graphical computation of the model's prediction for an arbitrary instance $x =< x_1, x_2 >$. For each input feature, we use the plotted function to map its value from the x-axis to that value's contribution on the y-axis. All that remains is to sum the two contributions and the expectation $E[f] = 4$.

To illustrate, observe the linear model $f(x_1, x_2) = 2x_1 - 3x_2 + 4$, with both input features uniformly distributed on $[-1, 1]$. How much do the two input features contribute for the prediction $f(\frac{1}{2}, \frac{1}{3})$? For this instance, the situational importance of the first and second feature are 1 and -1, respectively. Therefore, both features contribute positively. We can plot the average situational importance ψ of each value (for this additive model, all situational contributions of a particular value are the same; subsequently, so is the average situational contribution), to obtain an overview of how each feature contributes across all of its values (see Figure 1). This plot not only shows how different feature values contribute, it can also be used to semi-graphically compute the prediction for any instance. As previously mentioned, such a plot can be produced for any additive model, a fact that has been exploited in developing several model specific explanation methods [11, 9, 10, 13, 12]

Computing the contributions for our illustrative example was simple, because the model is known and the features do not interact (the model is additive). Therefore, the contribution of some x_i is the same across all instances, regardless of the values of other features. In our problem setting, however, the model is unknown and no assumptions are made other than that the model maps from some known input feature space to a known codomain. These restrictions are necessary for the method to be general, but restrict

us to changing the inputs and observing the outputs. Previous general approaches [27, 28] tackled the problem in the following way:

$$\varphi_i(x) = f(x_1, ..., x_n) - E[f(x_1, ..., X_i, ..., x_n)] \tag{2}$$

Eq. (2) is basically the difference between a prediction for an instance and the expected prediction for the same instance if the $i-$th feature had not been known. In practice, expression Eq. (2) is not difficult to approximate (or compute, if the feature's domain is finite) - we have to perturb the values of the $i-$th feature, while the values of other input features remain fixed, and then average the prediction. Additionally, if f is an additive model, Eq. (2) is equivalent to Eq. (1), so in the case of an additive model, we do not lose any of the previously mentioned advantages associated with explaining an additive model.

However, when the model is not additive, as is often the case in practical applications, the approach gives undesirable results. For example, observe the model $f(x_1, x_2) = x_1 \vee x_2$, where both features are uniformly distributed on $\{0, 1\}$. When computing the contribution of the first feature for $f(1, 1) = 1$, we see that perturbing its value does not change the prediction - the first features contribution is 0. The same holds for the second feature. Therefore, both features get a 0 contribution, which is undesirable.

We learn from the previous example that perturbing one feature at a time gives undesirable results and that all subsets of features have to be perturbed to avoid such issues. In our previous work we proposed a general method for explaining classification and, separately, regression models that dealt with the aforementioned shortcomings of existing general explanation methods [29, 30]. This paper builds on our previous work. We address the general explanation of both classification and regression models and provide a thorough empirical analysis of running times, illustrative examples, and the results of a user study of the usefulness of the contribution-based instance explanations. We also discuss two extensions that improve the algorithms efficiency and show that in the case of additive models, the proposed method is also equivalent to the explanation commonly used for additive models.

The remainder of the paper is organized as follows. In Section 2 we provide the essential background. In Section 3 we describe the approximation algorithms for computing a feature's contribution for an instance and the average contribution of a feature's value. Section 4 is dedicated to empirical results and visual inspection of instance and model visualizations. In Section

5 we describe a user study. With Section 6 we conclude the paper.

2 Computing a Feature's Contribution

The following notation will be used throughout the paper. Let $\mathcal{X} = [0,1]^n$ be our feature space, Y the target variable, and $\{y_i; x_{1,j}, x_{2,j}, ..., x_{n,j}\}_{i=1}^{M}$ a data set of M instances. The function $f : \mathcal{X} \to \Re$ represents the model that is used to predict the value of the target variable for an instance $x \in \mathcal{X}$.

First, let us observe how a feature's value contributes for a simple linear model. That is, let us assume for a moment that f takes the form

$$f(x) = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n.$$

The contribution of the i-th feature's value for some instance $x \in \mathcal{X}$ is the difference between the model's prediction and the expected prediction if the i-th feature's value is not known:

$$\begin{aligned} \varphi_i^{additive}(x) &= \beta_0 + ... + \beta_i x_i + ... + \beta_n x_n - (\beta_0 + ... + \beta_i \mathbb{E}[x_{,i}] + ... + \beta_n x_{,n}) \\ &= \beta_i (x_i - E[X_i]). \end{aligned} \tag{3}$$

The contribution in Eq. (3) is sometimes also referred to as the situational importance of x_i [26]. Observe how such a contribution is independent of the values of other features. This is due to the fact that the linear model is additive (that is, the features do not interact). This property makes the linear model and other additive models easy to interpret.

However, in practice we are often not dealing with an additive model, so any subset of features might interact. Therefore, to avoid the shortcomings of previous general methods, we have to observe the contribution of each subset of feature values. For this purpose, Eq. (3) has to be generalized. First, we define the model's prediction conditional to only a subset of features' values being known:

$$f_Q(x) = \mathbb{E}[f|X_i = x_i, \forall i \in Q], \tag{4}$$

where $Q \subseteq S = \{1, 2, ..., n\}$ represents a subset of features. Eq. (4) allows us to define the contribution of a subset of feature values:

$$\Delta_Q(x) = f_Q(x) - f_{\{\}}(x). \tag{5}$$

Eq. (5) can be interpreted as the change in prediction caused by observing the values of a certain subset of features for some instance $x \in \mathcal{X}$. The number of Δ-terms grows exponentially with the number of features. To provide a contribution similar to the one for the linear model, we have to map these 2^n terms into n contributions, one for each feature's value. First, we implicitly define interactions by saying that the contribution of a subset of feature values is the sum of all interactions across all subsets of those feature values:

$$\Delta_Q(x) = \sum_{W \subseteq Q} \mathcal{I}_W(x). \tag{6}$$

Which uniquely defines the interactions:

$$\mathcal{I}_Q(x) = \Delta_Q(x) - \sum_{W \subset Q} \mathcal{I}_W(x). \tag{7}$$

Finally, each interaction is divided among the participating feature values, which defines the contribution:

$$\varphi_i(x) = \sum_{" \subseteq S \setminus \{i\}} \frac{\mathcal{I}_{W \cup \{i\}}(x)}{|W| + 1}. \tag{8}$$

This leads to the following explicit definition:

$$\varphi_i(x) = \sum_{Q \subseteq S \setminus \{i\}} \frac{|Q|!(|S| - |Q| - 1)!}{|S|!} (\Delta_{Q \cup \{i\}}(x) - \Delta_Q(x)). \tag{9}$$

Eq. (9) is the Shapley value [31]. As such, it has the following desirable properties.

- $\sum_{i=1}^{n} \varphi_i(x) = \Delta_S(x)$
- $\forall W \subseteq S \setminus \{i\} : \Delta_W = \Delta_{W \cup \{i\}} \Rightarrow \varphi_i(x) = 0$
- $\forall W \subseteq S \setminus \{i, j\} : \Delta_{W \cup \{i\}} = \Delta_{W \cup \{j\}} \Rightarrow \varphi_i(x) = \varphi_j(x)$

That is, the contributions are implicitly normalized, which makes them easier to interpret and compare. If a feature's value does not have any impact on the prediction, will be assigned a 0 contribution. And, if two features' values have the a symmetrical impact across all subsets, they will be assigned equal contributions.

2.1 Illustrative example

It follows from the equivalence with the Shapley value that the contributions computed with the described method will not have the same problem as previous general methods. We illustrate this and the method itself with the following example. Let $\mathcal{X} = [0,1]^3$ and

$$f(x) = \begin{cases} 1 & \text{if}(x_1 > 0.5) \vee (x_2 > 0.5) \vee (x_3 > 0.5), \\ 0 & \text{otherwise.} \end{cases}$$

We assume that all instances from $\mathcal{X}$ are equally probable. Let us compute the contributions for the prediction $f(0.9, 0.7, 0.3) = 1$. This example is similar to the disjunction example from the Introduction - existing methods would assign all three features a zero contribution.

All conditional predictions, with the exception of $E[f] = \frac{7}{8}$ and $E[f|x_3 = 0.3] = \frac{6}{8}$, equal 1, because it is enough that one of the input features satisfies the condition. The Δ terms are $\Delta_{\{3\}} = -\frac{1}{8}$ and $\Delta_{\{1\}} = \Delta_{\{2\}} = \Delta_{\{1,2\}} = \Delta_{\{1,3\}} = \Delta_{\{2,3\}} = \Delta_{\{1,2,3\}} = +\frac{1}{8}$.

Interactions of input features (with themselves) are non-zero ($\mathcal{I}_{\{1\}} = \mathcal{I}_{\{2\}} = \frac{1}{8}$ and $\mathcal{I}_{\{3\}} = -\frac{1}{8}$), because each feature contributes by itself (as opposed to, for example, xor, where relevant features would not contribute anything by themselves. Interactions of pairs of features are $\mathcal{I}_{\{1,3\}} = \mathcal{I}_{\{2,3\}} = \frac{1}{8}$ and $\mathcal{I}_{\{1,2\}} = -\frac{1}{8}$. We encounter a negative interaction of two or more features, because the first two features contribute less together than the sum of their individual predictions. The interaction of all three is also negative $\mathcal{I}_{\{1,2,3\}} = \Delta_{\{1,2,3\}} - \sum_{Q \subset \{1,2,3\}} \mathcal{I}_Q = -\frac{1}{8}$.

All that remains is to divide the interactions among the participating features and form the contributions (see Eq. (8)). The contribution of the third input feature is $\varphi_3 = \mathcal{I}_{\{3\}} + \frac{\mathcal{I}_{\{1,3\}} + \mathcal{I}_{\{2,3\}}}{2} + \frac{\mathcal{I}_{\{1,2,3\}}}{3} = -\frac{1}{24}$. Due to symmetry, the contributions of the first two input features will be the same. Therefore, we only compute the first: $\varphi_1 = \mathcal{I}_{\{1\}} + \frac{\mathcal{I}_{\{1,2\}} + \mathcal{I}_{\{1,3\}}}{2} + \frac{\mathcal{I}_{\{1,2,3\}}}{3} = \frac{1}{12} = \varphi_2$. The contributions are non-zero and in accordance with our intuition the first two

features have a positive contribution, while the third input feature has a negative contribution.

3 Approximation Algorithm

The biggest issue with Eq. (9) is that it is infeasible for practical use. The following approximation is used to reduce the computational complexity. We start by writing a different but equivalent formulation of Eq. (9):

$$\varphi_i(x) = \frac{1}{n!} \sum_{\mathcal{O} \in \pi(N)} \left(\Delta_{Pre^i(\mathcal{O}) \cup \{i\}} - \Delta_{Pre^i(\mathcal{O})} \right), \quad i = 1, ..., n, \tag{10}$$

where $\pi(n)$ is the set of all ordered permutations of the feature indices $\{1, 2, ..., n\}$ and $Pre^i(\mathcal{O})$ the set of all indices that precede i in permutation $\mathcal{O} \in \pi(n)$.

If the cost of computing the Δ-terms would be zero, Eq. (10) could be approximated using a simple sampling algorithm, where $\left(\Delta_{Pre^i(\mathcal{O}) \cup \{i\}} - \Delta_{Pre^i(\mathcal{O})} \right)$ would be one sample (see, for example, [32]). However, the computational complexity of computing the Δ-terms is exponential. As shown in [30], it is sufficient to limit ourselves to such distributions of instances p that individual features are distributed independently. Now Eq. (5) can be simplified into:

$$\begin{aligned} \Delta_Q(x) &= f_Q(x) - f_{\{\}}(x) = \\ &= \sum_{w \in \mathcal{X}; \forall i: (w_i = x_i \vee i \notin S)} p(w) f(w) - \sum_{w \in \mathcal{X}; \forall i: (x_i = a_i)} p(\vec{x}) f(w) = \\ &= \sum_{w \in \mathcal{X}} p(w) (f(w_{[w_i = x_i, i \in S]}) - f(w), \end{aligned} \tag{11}$$

where the notation $w_{[w_i = x_i, i \in S]}$ denotes instance w with the value of feature i replaced with that feature's value in instance x, for each $i \in S$. For example, with $\vec{w} = \langle 2, 4, 6 \rangle$ and $\vec{x} = \langle 3, 5, 7 \rangle$, $\vec{w}_{[w_i = x_i, i \in \{1,3\}]} = \langle 3, 4, 7 \rangle$.

The Δ-terms in Eq. (10) are replaced with Eq. (11) to obtain:

$$\varphi_i(x) = \frac{1}{n!} \sum_{\mathcal{O} \in \pi(N)} \sum_{\vec{x} \in \mathcal{X}} p(\vec{x}) \cdot (f(\vec{x}_{[x_j = a_j, j \in Pre^i(\mathcal{O}) \cup \{i\}]}) - f(\vec{x}_{[x_j = a_j, j \in Pre^i(\mathcal{O})]})), \tag{12}$$

The following sampling procedure is used [29]. Let

$$V_{\mathcal{O},\vec{x}\in\mathcal{X}} = (f(\vec{x}_{[x_j=a_j, j\in Pre^i(\mathcal{O})\cup\{i\}]}) - f(\vec{x}_{[x_j=a_j, j\in Pre^i(\mathcal{O})]})),$$

all permutation / instance pairs, be the sampling population. When sampling at random and with replacement, we draw sample $V_{\mathcal{O},\vec{x}\in\mathcal{X}}$ with probability $p(z)$. Let us draw m such samples $V_1, V_2, ..., V_m$ at random with replacement and define:

$$\hat{\varphi}_i = \frac{1}{m}\sum_{j=1}^{m} V_j, \tag{13}$$

If follows that $\hat{\varphi}_i$ is approximately normally distributed with mean φ_i and variance $\frac{\sigma_i^2}{m}$, where σ_i^2 is the population variance. That is, $\hat{\varphi}_i$ is an unbiased and consistent estimator of $\varphi_i(x)$. The described approximation algorithm is summarized in Algorithm 1.

Algorithm 1 Approximating the i-th features contribution for model f, instance $x \in \mathcal{X}$ and distribution p. Draw m samples.

$\varphi_i(x) \leftarrow 0$
for 1 to m **do**
  select, at random, permutation $\mathcal{O} \in \pi(n)$
  select, at random, $z \in \mathcal{X}$
  construct two instances:

$\vec{b}_1 \leftarrow$ [preceding i-th in $\mathcal{O}$] [i] [succeeding i-th in $\mathcal{O}$] (take values from x / take values from z)

$\vec{b}_2 \leftarrow$ [preceding i-th in $\mathcal{O}$] [i] [succeeding i-tega v $\mathcal{O}$] (take values from x / take values from z)

  $\varphi_i(x) \leftarrow \varphi_i(x) + f(\vec{b}_1) - f(\vec{b}_2)$

end for
$\varphi_i(x) \leftarrow \frac{\varphi_i(x)}{m}$

3.1 Quasi-Random and Adaptive Sampling

In this section, we discuss two extensions that can be used to increase the efficiency of the described approximation algorithm. First, the approximation algorithm described in the previous section can be considered a form of Monte Carlo integration. Therefore, for faster convergence, quasi-random sampling can be used instead of pseudo-random sampling [33, 34]. In our experiments, we used the Sobol low-discrepancy quasi-random sequence [35].

And second, to compute the explanation for an instance x, we need to compute the contribution for each of the n features for that instance. In practice, we want to do this in a controlled amount of time to minimize the overall approximation error. The approximation error of the estimator $\varphi_i(x)$ depends on the population variance which may not be the same for all features. Given that in practice, we are limited to a certain number of samples m, it makes sense to adapt m_i the number of samples drawn for a feature to that feature's variance σ_i^2. We discuss two cases - minimizing the squared $\sum_{i=1}^{n}(\hat{\varphi}_i - \varphi_i)^2$ and the absolute approximation error $\sum_{i=1}^{n}|\hat{\varphi}_i - \varphi_i|$.

Recall that the estimate $\hat{\varphi}_i$ is approximately normally distributed $\hat{\varphi}_i \approx N(\varphi_i, \frac{\sigma_i^2}{m_i})$. It follows that $\hat{\varphi}_i - \varphi_i \approx N(0, \frac{\sigma_i^2}{m_i})$. The distribution of the absolute error for the i-th feature $Z_i = |\hat{\varphi}_i - \varphi_i|$ is half-normal, with $E[Z_i] = \sqrt{\frac{\sigma_i^2}{m_i}}\sqrt{\frac{2}{\pi}}$. The expectation for the sum of absolute errors is

$$E\left[\sum_{i=1}^{n} Z_i\right] = \sum_{i=1}^{n} n\sqrt{\frac{\sigma_i^2}{m_i}}\sqrt{\frac{2}{\pi}} = \sqrt{\frac{2}{\pi}}\sum_{i=1}^{n}\frac{\sigma_i}{\sqrt{m_i}}.$$

Similarly, for the sum of squared errors, we take $Z_i \approx (\hat{\varphi}_i - \varphi_i)^2$. The expectation $E[Z_i^2] = Cov[Z_i, Z_i] + 2E[Z_i] = \frac{\sigma_i^2}{m_i}$. The expectation for the sum of absolute errors is

$$E\left[\sum_{i=1}^{n} Z_i\right] = \sum_{i=1}^{n}\frac{\sigma_i^2}{m_i}.$$

In practice, the variances σ_i are not known beforehand, so it is necessary to take samples for each input feature to estimate the variance. The minimum number of samples is a tradeoff between the risk of wasting samples on a less relevant input feature and the risk of missing an important feature.

After the minimum amounts of samples have been taken, the goal is to distribute m_{max}, the total number of samples we can compute, among indi-

vidual features in such a way that we minimize the expected error. Regardless of which error we use, a greedy approach is optimal. That is, if the current amount of samples taken for each feature are $m_1, ..., m_n$ then we should take the sample for the feature that maximizes $\sqrt{\frac{\sigma_i^2}{m_i}} - \sqrt{\frac{\sigma_i^2}{m_i+1}}$ (or $\frac{\sigma_i^2}{m_i} - \frac{\sigma_i^2}{m_i+1}$). This is a direct consequence of the fact that functions $g(z) = \frac{\sigma_i}{\sqrt{z}}$ and $g(z) = \frac{\sigma_i^2}{z}$ are both strictly decreasing on $z \in \Re^+$. Therefore, the currently best choice is also better than all possible future choices, regardless of the order in which future samples are taken.

Algorithm 2 Approximating all features' contributions for model f, instance $x \in \mathcal{X}$ and distribution p. Draw m_{min} samples for each feature, draw a total of $m_{max} >= n \cdot m_{min}$ samples.

for $i = 1$ to n **do**
  $m_i \leftarrow 0$, $\varphi_i(x) \leftarrow 0$
end for
while $\sum_{i=1}^{n} m_i < m_{max}$ **do**
  if $\forall i : m_i \geq m_{min}$ **then**
    pick a $j \in \{1..n\}$ which maximizes $(\sqrt{\frac{\sigma_j^2}{m_j}} - \sqrt{\frac{\sigma_j^2}{m_j+1}})^\dagger$
  else
    pick a j, such that $m_j < m_{min}$
  end if
  $\varphi_i(x) \leftarrow \varphi_i(x)+$ result of Algorithm 1 for j-th feature and $m = 1$
  Update σ_i^2 using an incremental algorithm.
end while
for $i = 1$ to n **do**
  $\varphi_i(\vec{a}) \leftarrow \frac{\varphi_i(\vec{a})}{m_i}$
end for

† if minimizing the quared error, maximize $(\frac{\sigma_j^2}{m_j} - \frac{\sigma_j^2}{m_j+1})$ instead

The adaptive sampling version of the algorithm is summarized in Algorithm 2. Note that we used Knuth's incremental algorithm for computing the variance [36, 37].

3.2 Average Contribution of a Feature's Value

The plots mentioned in the introduction are a common and efficient way of presenting an overview of additive models. Such plots show, for each feature, the function that maps its values to situational contributions of those values. Recall that for additive models, where the features do not interact, the situational contribution of the i-th features j-th value is the same for all instances. However, if the model is not additive and the features interact, then the situational contribution of a features value depends on the values of other features.

Therefore, to produce a similar plot with the proposed method, we average the value's contributions across all instances with that value:

$$\begin{aligned}
\psi_{i,j} = &\sum_{x\in\mathcal{X}, x_i=\mathcal{X}_{i,j}} p(x)\varphi_i(x) = \sum_{x\in\mathcal{X}} p(x)\varphi_i(x_{[x_i=\mathcal{X}_{i,j}]}) = \\
&= \sum_{x\in\mathcal{X}} p(x) \left(\frac{1}{n!} \sum_{\mathcal{O}\in\pi(N)} \sum_{z\in\mathcal{X}} p(z)(f(z_{[z_j=x_j, j\in Pre^i(\mathcal{O}); z_i=\mathcal{X}_{i,j}]}) - f(z_{[z_j=x_j, j\in Pre^i(\mathcal{O})]})) \right) = \\
&= \frac{1}{n!} \sum_{\mathcal{O}\in\pi(N)} \left(\sum_{x\in\mathcal{X}} p(x) \sum_{z\in\mathcal{X}} p(z_{[z_j=x_j, j\in Pre^i(\mathcal{O}); z_i=\mathcal{X}_{i,j}]}) - f(z_{[z_j=x_j, j\in Pre^i(\mathcal{O})]})) \right) = \\
&= \frac{1}{n!} \sum_{\mathcal{O}\in\pi(N)} \sum_{z\in\mathcal{X}} p(z) \left(f(z') - f(z) \right) \\
&= \sum_{z\in\mathcal{X}} p(z) \left(f(z') - f(z) \right),
\end{aligned} \tag{14}$$

where $\vec{x}'$ ($\vec{z}'$) is a vector, which equals $\vec{x}$ ($\vec{z}$), with the exception of the i-th component, which is set to $\mathcal{X}_{i,j}$.

Let $\psi_i(x), i = 1..n$ be the average contribution functions. If f is additive then it holds for each input feature i and its value x that $\psi_i(x) = f_i(x) - E[f_i]$:

$$\begin{aligned}
\psi_i(x) &= \sum_{\vec{z}\in\mathcal{X}} p(\vec{z})(f(\vec{z}_{z_i=x}) - f(\vec{z})) = \\
&= \sum_{\vec{z}\in\mathcal{X}} p(\vec{z})(f_i(x) - f_i(z_i)) =
\end{aligned}$$

$$= \sum_{\vec{z} \in \mathcal{X}} p(\vec{z}) f_i(x) - \sum_{\vec{z} \in \mathcal{X}} p(\vec{z}) f_i(z_i) =$$
$$= f_i(x) - E[f_i].$$

That is, in the case of an additive model, the average contribution of a feature's value equals the situational contribution of that value.

4 Experimental Results

The experiments are divided into two parts. First, we analyze the running times across several different types of models and data sets. And second, we illustrate the method's practical usefulness by visually inspecting several instance and model explanations.

The list of artificial data sets that we used in our experiments is shown in Table 1. These data sets were constructed for the purpose of experimental verification of how general explanation methods perform on data with concepts such as disjunction, xor, conditionally independent features redundant, and random features. For most data sets only a brief description is given and further details can be found in [27, 29, 30]. For those artificial data sets that are visualized in the second part of this section a more detailed description will be given.

We also included in the experiments several well-known regression and classification data sets: autoMpg, bodyfat, concrete, elevators, fishcatch, fruitfly, housing, machinecpu, pollution, stock, wine, and wisconsin (regression), anneal, breastCancerLJ, hepatitis, iris, monks1, monks2, monks3, mushroom, nursery, soybean, and zoo (classification). The data sets are available in *.arff* format from the Weka website (`http://www.cs.waikato.ac.nz/ml/weka/`). Most can also be found at the UCI Machine Learning repository [38].

We included ten different variations of learning algorithms for classification and seven different variations for regression (see Table 2). All used learning algorithms were from the Weka [39] machine learning software. Unless otherwise noted, default settings were used.

4.1 Running Times Analysis

With the first experiment we illustrate the benefits of using adaptive sampling and/or quasi-random sampling. We included all regression data set/regression

Table 1: Number of instances (#I), total number of input features (#F), and brief description of artificial data sets.

	Name	#I	#F	Description
Classification	cChess	2000	4	Color of 4x4 chessboard point.
	cCondInd	2000	8	Conditionally independent features.
	cCross	2000	6	Even or odd quadrant in coordinate system.
	cDisjunctB	2000	5	Disjunction with binary input features.
	cDisjunctN	2000	5	Disjunction with numeric features.
	cGroup	2000	4	Clusters.
	cRandom	2000	4	Random input features.
	cRedundant	2000	5	Disjunction with redundant features.
	cSphere	2000	5	Point lies in the interior of a sphere.
	cXor	2000	6	Xor.
Regression	rDisjunctB	2000	5	Disjunction with binary input features.
	rDisjunctN	2000	5	Disjunction with numeric features.
	rLinear	2000	5	Linear problem.
	rLinNoisy	2000	5	Linear problem with noise.
	rLocLinear	2000	5	Locally linear problem.
	rNonLinPoly	2000	5	Third degree polynomial.
	rNonLinTrig	2000	5	Trigonometric function.
	rRandom	2000	4	Random input features.
	rRedundant	2000	5	Disjunction with redundant features.
	rXor	2000	6	Xor.

Table 2: A list of learning algorithms that were included in the experiments. IBk and MultilayerPerceptron were used for both regression and classification.

Name	Description
AdaBoostM1	Boosting with Naive Bayes or decision tree as base learner.
Bagging	Bagging with either decision tree or regression tree as base learner.
IBk	k-Nearest Neighbors with either k = 1 or k = 11.)
J48	Decision tree.
LinearRegression	Linear regression.
Logistic	Logistic regression.
M5P	Regression tree.
MultilayerPerceptron	Multi-layer artificial neural network with on hidden level.
NaiveBayes	Naive Bayes classifier.
SVO	Support Vector Machine with second degree polynomial kernel.
SVMreg	Regression SVM.

learner and classification data set/classifier pairs in this experiment. For each such pair, we trained the model and computed the mean squared approximation error across all instances. We computed the error at different amounts of samples per features and for all four combination of the basic approximation algorithm and for enhancements (both, just quasi-random, just adaptive, neither). The results, shown in Figure 2, suggest that both enhancements improve the efficiency of the algorithm. That is, on average, fewer samples are needed to achieve the same approximation error. The improvement achieved with quasi-random sampling is small, compared to the improvement achieved by adaptive sampling. Best results are achieved when both enhancement are used.

To better understand how many features is still a reasonable number to generate an explanation for, we generated additional data sets. The data set *linear50* contains 1000 instances with 50 standardized numerical input features. The class value is a linear combination of features. Odd features are irrelevant while the integer coefficients for other features were chosen at random from -5 to 5. The *datgen40* data set contains 1000 instances described with 40 nominal features (30 of which are unrelated to the class value) and each feature has 10 values. This is a classification data set with 10 classes (see A for a detailed description). Note that this data set was created using Melli's data set generator [40].

We ran the following experiment for each of the two data sets separately and all appropriate learning algorithms. Starting with the first input feature

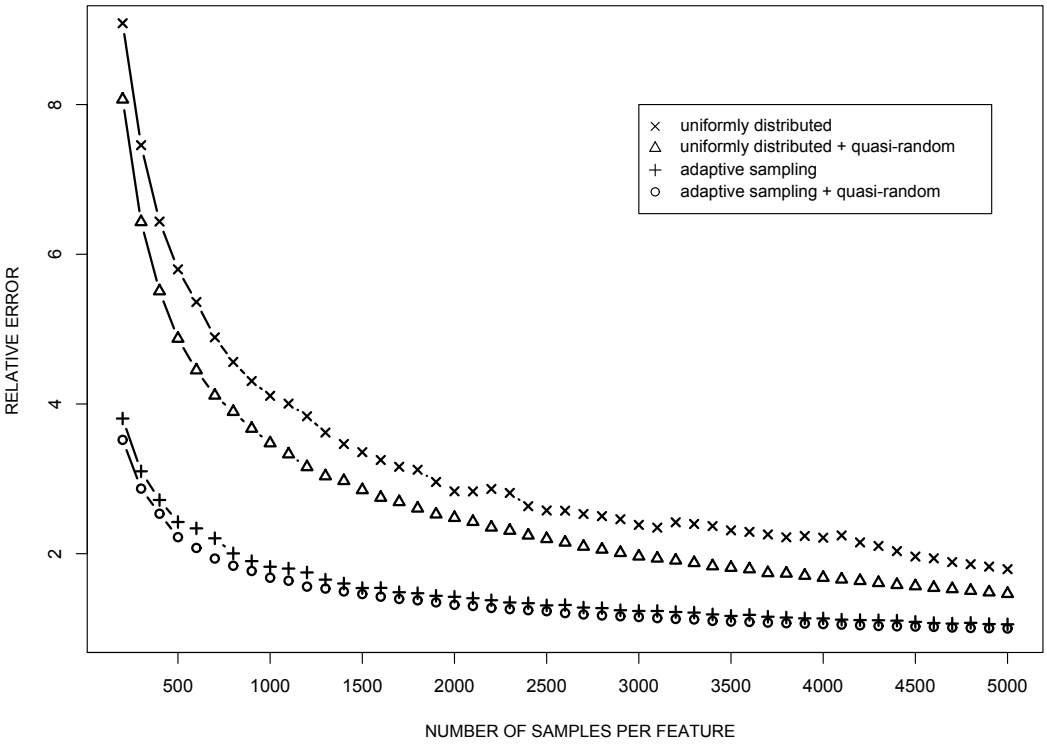
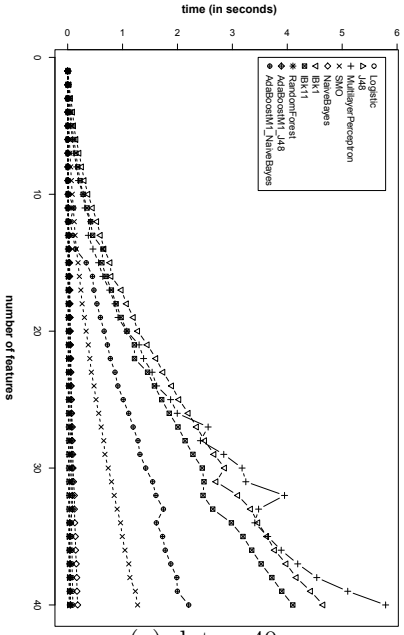


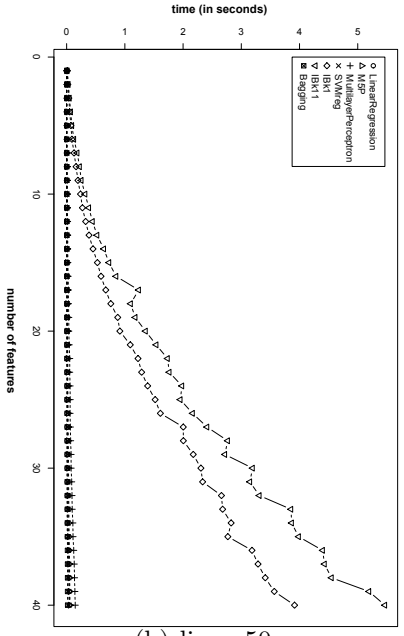
Figure 2: Relative approximation against number of samples per feature for four variants of the approximation algorithm. Errors are relative to the error at (5000 $\times$ number of input features) samples with both enhancements.

and then incrementally adding features, we measured the time required to compute all contributions for a single instance. Note that the number of samples taken m_{max} was such that the probability of having a relative approximation error of more than 1% was 5% (relative to the absolute value of the contribution). Adaptive sampling was used, but not quasi-random sampling.

The results shown in Figure 3 suggest that contributions can be computed in real-time for a few dozen features, regardless of the choice of model. For some models, this number can be extended to over a hundred features. The differences between models are in part a consequence of different variances but mostly due to the differences in the time complexity of computing a single prediction, which is the key component of the approximation algorithms time complexity. The nearest neighbors model stands out, because the prediction



(a) datgen40



(b) linear50

Figure 3: Running times for computing the contributions for one instance.

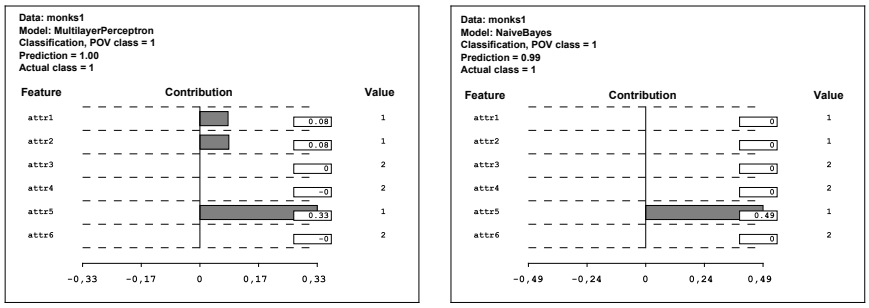
involves the computation of all the distances, without optimization, such as the use of kd-trees. Note that the experiments were run on an off-the-shelf computer with a 2.4GHz CPU and 2GB of RAM.

4.2 Instance Visualization

To visualize the results (feature contributions) of the proposed explanation method, we use two types of visualizations: instance visualizations and model visualizations. The latter are discussed later, while the former is, as the name suggests, a visualization of the features' contributions φ for a particular instance. Figure 4 shows a pair of such visualizations for the same instance from the Monks1 data set but for two different types of models. At the top of an instance visualization we can find basic information: the name of the data set, the model who's prediction the contributions are for, the point-of-view class (classification only), the model's prediction for this instance, and the actual (correct) class value. On the left hand side, the names of the features are listed and on right hand side, the values of the features for this instance. The boxes contain the features' contributions for this instance . These contributions are also plotted as bars to simplify comparison and identification of features with the largest contribution.

An instance visualization reveals how individual features contribute to the model's prediction for that instance. For example, for the Monks1 data set, the class value is 1 if (attr1 = attr3 $\vee$ attr 5 = 1) and 0 otherwise. The pair of instance visualizations for the same instance from Monks1 but two different models trained on this data set provide us with additional information about how the features influence the models' prediction (see Figure 4). Although the models are of a different type, the general method facilitates comparison and reveals an important difference between the two models.

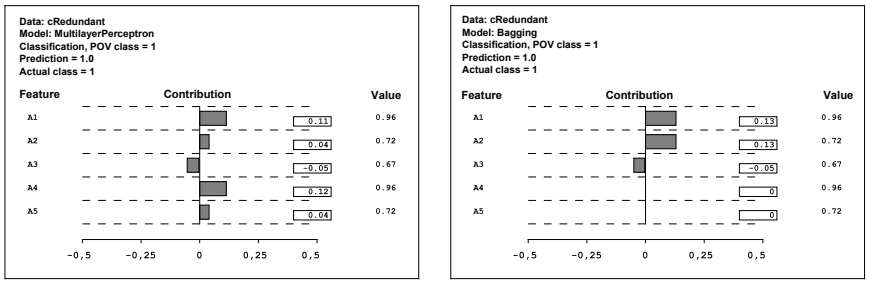
The second pair of instance visualizations is for two different types of models trained on the cRedundant data set (see Figure 5. This data set has 5 numerical input features. The class value equals 1 if $A_1 > 0.5$, $A_2 > 0.7$ or $A_3 < 0.4$. Otherwise it is 0. Note that the remaining two features A_4 and A_5 are copies of A_1 and A_2, respectively, which introduces some redundancy. For this instance, the values of the first three input features are 0.96, 0.72, and 0.67. The first two features satisfy the condition, while the third does not. Given that both models are successful predictors for this data set, they have learnt these concepts and, appropriately, the first two features are assigned a positive contribution, while the third contributes against the class



(a) (b)

Figure 4: The Naivni Bayes model, due to its assumptions of conditional independence of input features, can not model the importance of the equivalence between attr1 and attr2 (both have a zero contribution). Despite this limitation, it correctly predicts the class value, because for this instance, attr5 = 1 is sufficient (this feature has a substantial positive contribution). The artificial neural network correctly models both concepts.

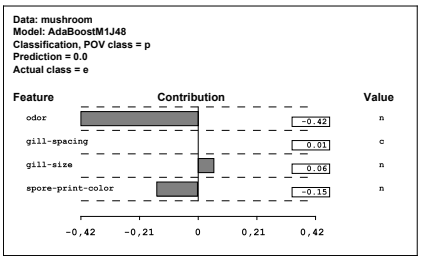
value being 1. Note that the artificial neural network takes into account redundant features as well, while bagging with decision trees only takes into account one of each of the input features redundant copies. Although both models are equally good predictors, the explanations are different, because the explanations reveal what the models have learnt.



(a) (b)

Figure 5: Two instance visualizations for the same instance from the cRedundant data set and two different models.

The final pair of instance visualizations focus on real-world examples (see Figure 6).



(a) (b)

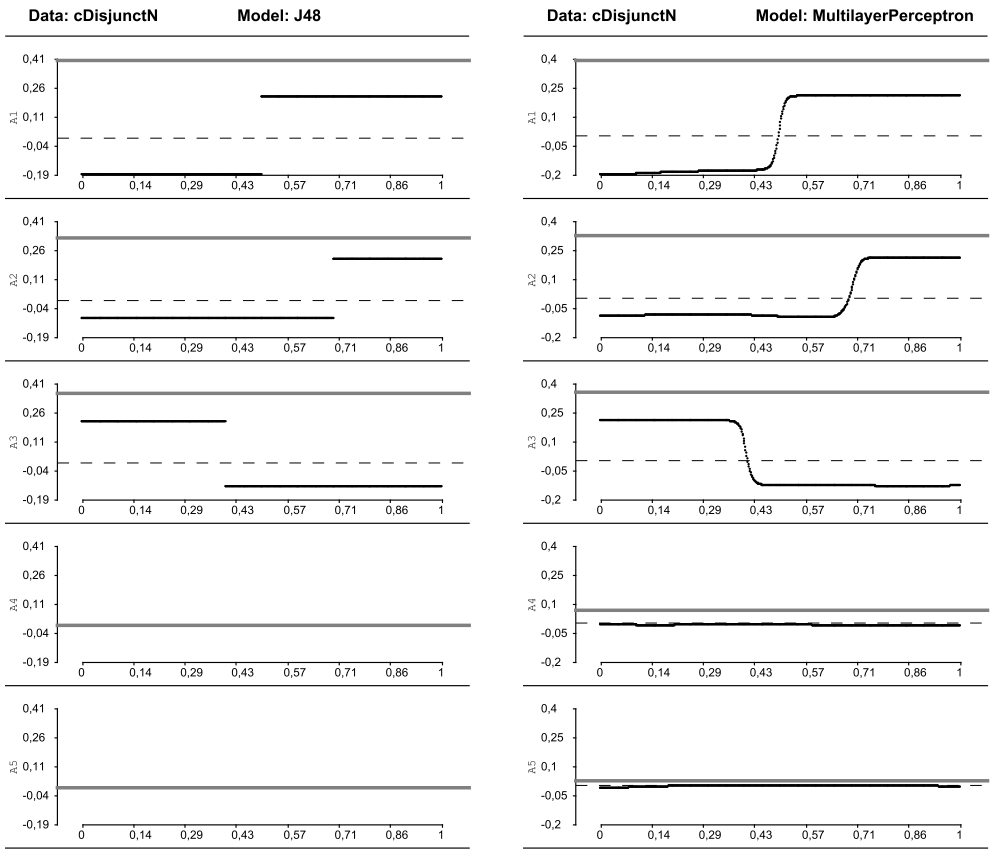
Figure 6: The boosting with decision trees model predicts that this mushroom from the mushroom data set is not poisonous. The contributions reveal that (odor = n [none]) contributes strongly against the mushroom being poisonous. (spore print color = n [brown]) also speaks against poisonous, while narrow gill size weakly contributes towards the mushroom being poisonous.

4.3 Model Visualization

The other type of visualization is the model visualization (see, for example, 7(a). The model visualization is in fact the visualization of average contributions of values for each feature (see 3.2). For each feature, the average contributions are plotted against that features value (black line). The importance of each feature (the standard deviation of its contributions) is also included in the form of a gray line. Similar to instance visualizations, the name of the data set and model are shown at the top of the visualization.

The model visualization provides us with an overview of how features contribute to the model's predictions. For example, observe Figure 7(a) - the model visualization of the decision tree that was trained on the cDisjunctN data set and is good at predicting the class values. The cDisjunct data set is similar to the cRedundant data set, however, the fourth and fifth feature are not copies of the first and second feature. Instead, they are unrelated to the class value. First, the model visualization allows us to quickly identify the most important features. In our example, the first three features have an equally high importance (gray line), while the remaining two features are (correctly) identified as irrelevant or of very little importance. And second, the plots provide additional information about how each feature contributes to the model. For example, the first feature (A1) has a negative contribution (speaks against class value 1) if its value is less than 0.5, but contributes positively, if its value is greater than 0.5. Also note that, as shown in Secion

3.2, if the model is additive, the the plot can also be use to graphically compute the prediction for an arbitrary instance form the data set.



(a) (b)

Figure 7: Model visualizations for two different models and the cDisjunctN data set. Both models learn the concepts behind the data and the plotted average contribution functions reveal where the individual features' contribution changes from negative to positive.

Due to the methods generality, model visualizations of different models or types of models can easily be compared. For example, Figure 7(b) is a model visualization for an artificial neural network trained on the cDisjunct data set. While the performance of both models is similar with respect to prediction quality, the models are slightly different. The average contribution functions are smooth for the artificial neural network, but characteristically

discontinuous for the decision tree. The artificial neural network also slightly overfit the data as the fourth and fifth feature do slightly influence the models predictions.

The next model visualization is for the already discussed Monks1 data set. By assuming conditional independence, the Naive Bayes model does not capture the importance of the equivalence of attr1 and attr2 and the model visualization reveals that only the fifth feature influences the model (sse Figure 8(a)). Furthermore, the value (attr5 = 1) has a positive contribution, while all other values have a negative contribution.

The final model visualization (see Figure **??** gives us additional information about how the features influence the decision tree trained on the mushroom data set. Because the decision tree is a good model of the data, the contributions also help us understand the concepts behind the mushroom data. By far the most important input feature is mushroom odor (almond, anise or no odor at all, speak against the mushroom being poisonous, other values speak for), spore print color has a very small influence - green spore print color speaks for the mushroom being poisonous.

5 A User Study

The usefulness of explanation methods (that is, an increase in understanding or level of trust) for end-users is usually evaluated through illustrative examples or by applying the method to a real-world problem. In the latter case, the resulting explanations are usually evaluated by experts from the field of the real-world problem. Therefore, in most cases, such an evaluation is very subjective. Only a few studies approach the evaluation in a more objective way. Huysmans et. al. [41] compared the usefulness of decision tables, binary decision trees, and decision rules in a study that included 51 post-graduate students. Provided with these different representations of knowledge, the students had to perform various understanding and prediction tasks. The authors measured prediction accuracy, speed of response, and the level of trust, and concluded that decision tables were most useful. Similar studies focused only on decision trees [42] and decision trees and decision rules [43].

Similar to [41] we decided to measure the effect of the explanation through the users' performance at prediction tasks. We performed a user study that involved 122 computer science students. The students were given a set of instances with correct class values from which to learn about the concepts

Table 3:

	T1				T2			
instance	A1	A2	A3	C	A1	A2	A3	C
learn1	11.76	70	12	52	A	450	21	27
φ	0	0	-60	-	-31	+11	0	-
learn2	11.56	55	16	82	A	250	21	2
φ	0	0	-30	-	-25	-20	0	-
learn3	11.86	32	22	127	B	280	22	12
φ	0	0	+15	-	-7	-28	0	-
learn4	12.12	80	12	60	B	600	20	70
φ	0	0	-52	-	+3	+20	0	-
learn5	12.31	74	28	172	B	800	22	75
φ	0	0	+60	-	+4	+24	0	-
learn6	11.65	44	21	120	C	250	21	56
φ	0	0	+8	-	+31	-22	0	-
learn7	11.34	72	25	150	C	200	19	40
φ	0	0	+38	-	+28	-35	0	-
test1	11.91	59	21	120	A	600	21	30
test2	11.87	28	29	180	B	100	23	94
test3	12.00	54	27	165	C	400	20	2
test4	11.73	33	17	90	C	800	20	100

behind the data (learning instances) and were then asked to make predictions for a set of instances without the correct class value. We hypothesized that the explanation in the form of feature contributions was made available for the learning instances it would give the users a better understanding of the concepts behind the data and the users would subsequently make better predictions.

For these purposes we constructed two sets of learning and testing instances instances, T1 and T2 (see Table 3). Each set consists of 7 learning instances and 4 test instances without class values. Both sets of instances have a real-world background. For T1, the input features and class value are cricket length, humidity, temperature, and number of chirps per minute, respectively. The latter is a linear function of temperature, while the remaining two input features are not relevant. For T2, the input features and class value are insecticide type, insecticide amount (in ml), temperature, and percentage of insects killed. The latter depends on insecticide type (C is stronger than B is stronger than A) and amount, while temperature is irrelevant.

The real-world examples were selected to make the problem less abstract and easier to relate to. Of course, the relationships between the input features

and class value were not revealed to the participants.

5.1 Experiment 1

In the first experiment each participant solved 2 tests and had 8 minutes available for each test. One half of the participants received T1 without explanations and the T1 with contributions while the other half received T2 without and then T2 with explanation. each participant received two tests. Note that all 56 participants were first-year students, which are assumed to have no substantial experience with knowledge discovery.

For each test instance separately, we ranked the mean squared errors of the participants predictions. We prefer ranks to actual mean squared errors to avoid the effect of outliers and facilitate comparison across all four test instances. For the group of 28 participants that received test T1, the average ranks were 36 (without contributions) and 20 (with contributions). For T2, the average ranks were 29.5 (without contributions) and 26.5 (with contributions). We used a Wilcoxon paired test to test the statistical significance and obtained the P-values 2.2×10^{-16} and 2.3×10^{-4} for T1 and T2, respectively. Therefore, for both tests, the contributions improved the user's predictions.

5.2 Experiment 2

Two groups of students participated in the second experiment: 52 first year students (group A) and 14 fourth year students with experience in data mining and knowledge discovery (group B). One half of each group received T1 without contributions and T2 with contributions, while the other half solved T2 without and T1 with contributions. Therefore, the task was assumed to be more difficult than in experiment 1. Instead of solving the same problem without and then with contributions, each participant solved one problem without contributions and then the other problem with contributions. Again, participants had 8 minutes for each set. Note that all of the participants were different from those in experiment 1.

Results were analyzed for each group separately and in the same way as in experiment 1. However, for experiment 1 the samples were not paired.

The results are shown in Table 4. Similar to experiment 1, we conclude that the contributions improved the user's predictions.

Table 4: Rezultati preverjanja kvalitete napovedi.

	With	**Without**	**P-value**
EXP2, A, T1	30.75	21.25	6.4×10^{-6}
EXP2, A, T2	28.25	23.75	1.5×10^{-2}
EXP2, B, T1	7.75	6.62	4.9×10^{-2}
EXP2, B, T2	8.30	5.70	1.2×10^{-2}

Table 5:

	With	**Without**	**P-value**
EXP1, T1	2.58	2.89	2.5×10^{-4}
EXP1, T2	2.73	2.78	2.4×10^{-1}
EXP2, A, T1	2.68	2.98	7.5×10^{-3}
EXP2, A, T2	2.62	2.64	4.6×10^{-1}
EXP2, B, T1	2.96	2.96	6.1×10^{-1}
EXP2, B, T2	2.71	2.87	2.1×10^{-1}

5.3 Effects on the Users' Level of Trust

In parallel we also measured, for each prediction, the user's level of trust in the correctness of own predictions. The users selected from a four-level scale from 1 (very unsure) to 4 (very sure). Results of the experiment can be found in Table 5. Similar to the quality of predictions, the level of trust also increases when the contributions are available. However, most of the differences lack statistical significance.

6 Conclusion

We described a general method for computing the contributions of input features for a prediction. The method is simple to implement and can be applied to any regression or classification model. The contributions of a particular value can be averaged across all instances to an average contribution of that value that is the basis of a model visualization. For additive models, such a visualization is equivalent to existing additive model-specific methods and general explanation methods and thus generalizes these approaches. Results across several types of models and data sets show that the method is an

efficient and useful tool for visualizing models, comparing them, and identifying potential errors. The method can compute the explanation in real-time for several dozen input features. This is sufficient for practical use, because similar to all explanation methods, the comprehensibility of the explanation (from a user's point-of-view) decreases with an increasing number of input features.

We also performed a study in which the participants learned from instances with easy-to-understand input features and class values, but unknown relationships. The results show that users made better predictions for new instances if feature contributions were included in the instances reserved for learning. This holds for both the case where the user solved the problem from the same data set without and then with explanations and the case where the user solved a problem with explanations, without having seen and solved the same problem without explanations. While the users' confidence in the correctness of own predictions was at least as high if not higher when the feature contributions were available, the differences were not statistically significant. Therefore, we can not conclude that the explanations increase the users' confidence.

References

[1] I. Alvarez, Explaining the result of a decision tree to the end-user, in: Proceedings of the 16th European Conference on Artificial Intelligence, 2004, pp. 411–415.

[2] J. Blanchard, F. Guillet, H. Briand, Interactive visual exploration of association rules with rule-focusing methodology, Knowledge and Information Systems 13 (2007) 43–75.

[3] J. R. Cano, F. Herrera, M. Lozano, Evolutionary stratified training set selection for extracting classification rules with trade off precision-interpretability, Data & Knowledge Engineering 60 (1) (2007) 90–108.

[4] I. De Falco, A. Della Cioppa, A. Iazzetta, E. Tarantino, An evolutionary approach for automatically extracting intelligible classification rules, Knowledge and Information Systems 7 (2005) 179–201.

[5] S. Dehuri, S. Patnaik, A. Ghosh, R. Mall, Application of elitist multi-objective genetic algorithm for classification rule

generation, Applied Soft Computing 8 (1) (2008) 477–487. doi:http://dx.doi.org/10.1016/j.asoc.2007.02.009.

[6] Q. Shen, A. Chouchoulas, A rough-fuzzy approach for generating classification rules (2002).

[7] B. Becker, R. Kohavi, D. Sommerfield, Visualizing the simple Bayesian classier, in: KDD Workshop on Issues in the Integration of Data Mining and Data Visualization, 1997.

[8] I. Kononenko, Inductive and bayesian learning in medical diagnosis, Applied Artificial Intelligence 7 (1993) 317–337.

[9] M. Možina, J. Demšar, M. Kattan, B. Zupan, Nomograms for visualization of naive Bayesian classifier, in: PKDD 2004, Springer-Verlag, 2004, pp. 337–348.

[10] A. Jakulin, M. Možina, J. Demšar, I. Bratko, B. Zupan, Nomograms for visualizing support vector machines, in: KDD '05: 11th ACM SIGKDD, ACM, 2005, pp. 108–117.

[11] J. Lubsen, J. Pool, E. van der Does, A practical device for the application of a diagnostic or prognostic function, Methods of Information in Medicine 17 (1978) 127–129.

[12] M. W. Kattan, J. A. Eastham, A. M. Stapleton, T. M. Wheeler, P. T. Scardino, A preoperative nomogram for disease recurrence following radical prostatectomy for prostate cancer, Journal of the National Cancer Institute 90 (1998) 766–771.

[13] D. Szafron, B. Poulin, R. Eisner, P. Lu, R. Greiner, D. Wishart, A. Fyshe, B. Pearcy, C. Macdonell, J. Anvik, Visual explanation of evidence in additive classifiers, in: Proceedings of Innovative Applications of Artificial Intelligence, 2006.

[14] M. Holena, Extraction of logical rules from data by means of piecewise-linear neural networks, in: LNCS: Discovery Science, Springer, 2007.

[15] R. Krishnan, G. Sivakumar, P. Bhattacharya, Extracting decision trees from trained neural networks, Pattern Recognition 32 (1999) 1999–2009.

[16] R. Nayak, Generating rules with predicates, terms and variables from the pruned neural networks, Neural Networks 22 (4) (2009) 405–414.

[17] G. Towell, J. W. Shavlik, Extracting refined rules from knowledge-based neural networks, machine learning, Machine Learning 13 (1993) 71–101.

[18] D. S. Yeung, H. S. Fong, A knowledge matrix representation for a rule-mapped neural network, Neurocomputing 7 (1995) 123–144.

[19] N. H. Barakat, A. P. Bradley, Rule extraction from support vector machines: A sequential covering approach, IEEE Trans. on Knowl. and Data Eng. 19 (6) (2007) 729–741.

[20] L. Hamel, Visualization of support vector machines with unsupervised learning., in: Computational Intelligence in Bioinformatics and Computational Biology, IEEE, 2006, pp. 1–8.

[21] D. Martens, B. Baesens, T. Van Gestel, J. Vanthienen, Comprehensible credit scoring models using rule extraction from support vector machines, European Journal of Operational Research 183 (3) (2007) 1466–1476.

[22] F. Poulet, Svm and graphical algorithms: A cooperative approach, in: 4th IEEE ICDM, 2004, pp. 499–502.

[23] B. Ustn, W. Melssen, L. Buydens, Visualisation and interpretation of support vector regression models., Anal Chim Acta 595 (1-2) (2007) 299–309.

[24] N. A. Vien, N. H. Viet, T. Chung, H. Yu, S. Kim, B. H. Cho, Vrifa: A nonlinear SVM visualization tool using nomogram and localized radial basis function (LRBF) kernels, in: CIKM, 2009, pp. 2081–2082.

[25] Y. Zhang, H. Su, T. Jia, J. Chu, Rule extraction from trained support vector machines, in: PAKDD, 2005, pp. 61–70.

[26] C. H. Achen, Intepreting and Using Regression, Sage Publications, 1982.

[27] M. Robnik-Šikonja, I. Kononenko, Explaining classifications for individual instances, IEEE TKDE 20 (2008) 589–600.

[28] V. Lemaire, R. Fraud, N. Voisine, Contact personalization using a score understanding method, in: International Joint Conference on Neural Networks (IJCNN), 2008.

[29] E. Štrumbelj, I. Kononenko, An efficient explanation of individual classifications using game theory, Journal of Machine Learning Research 11 (2010) 1–18.

[30] E. Štrumbelj, I. Kononenko, A general method for visualizing and explaining black-box regression models, in: A. Dobnikar, U. Lotric, B. Ster (Eds.), ICANNGA (2), Vol. 6594 of Lecture Notes in Computer Science, Springer, 2011, pp. 21–30.

[31] L. S. Shapley, A Value for n-person Games, Vol. II of Contributions to the Theory of Games, Princeton University Press, 1953.

[32] J. Castro, D. Gómez, J. Tejada, Polynomial calculation of the shapley value based on sampling, Computers & Operations Research 36 (5) (2009) 1726 – 1730.

[33] H. Niederreiter, Low-discrepancy and low-dispersion sequences, Journal of Number Theory 30 (1) (1988) 51–70.

[34] H. Niederreiter, Random number generation and quasi-Monte Carlo methods, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1992.

[35] P. Jaeckel, Monte Carlo Methods in Finance, Wiley, New York, 2002.

[36] D. E. Knuth, The Art of Computer Programming, volume 2: Seminumerical Algorithms, Addison-Wesley, 1998.

[37] B. P. Welford, Note on a method for calculating corrected sums of squares and products, Technometrics 4(3) (1962) 419–420.

[38] A. Frank, A. Asuncion, UCI machine learning repository (2011).
URL `http://archive.ics.uci.edu/ml`

[39] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten, The weka data mining software: an update, SIGKDD Explor. Newsl. 11 (1) (2009) 10–18.

[40] G. Melli, The datgen dataset generator, `http://www.datasetgenerator.com`.

[41] J. Huysmans, K. Dejaeger, C. Mues, J. Vanthienen, B. Baesens, An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models, Decision Support Systems 51 (1) (2011) 141 – 154.

[42] B. Y. Lim, A. K. Dey, D. Avrahami, Why and why not explanations improve the intelligibility of context-aware intelligent systems, in: Proceedings of the 27th international conference on Human factors in computing systems, CHI '09, ACM, New York, NY, USA, 2009, pp. 2119–2128.

[43] H. Allahyari, N. Lavesson, User-oriented assessment of classification model understandability, in: Proceedings of the 11th Scandinavian Conference on Artificial Intelligence - SCAI 2011, 227, pp. 11–19.

A Description of datgen40 data set

$$(A_{15} = j \wedge A_{31} = f \wedge A_{33} = f) \vee (A_{33} = a) \Rightarrow C_1$$
$$(A_{12} = j \wedge A_{21} = f \wedge A_{33} = f) \vee (A_{33} = i) \Rightarrow C_2$$
$$(A_{12} = d \wedge A_{21} = f \wedge A_{33} = j) \vee (A_{01} = c \wedge A_{23} = h \wedge A_{01} = g) \vee (A_{33} = c \wedge A_{01} = g) \Rightarrow C_3$$
$$(A_{12} = j \wedge A_{23} = i \wedge A_{33} = b) \Rightarrow C_4$$
$$(A_{12} = j \wedge A_{01} = b \wedge A_{33} = d) \Rightarrow C_5$$
$$(A_{03} = g \wedge A_{21} = j \wedge A_{33} = j) \Rightarrow C_6$$
$$(A_{04} = h \wedge A_{23} = a \wedge A_{33} = b) \vee (A_{33} = f \wedge A_{31} = b) \Rightarrow C_7$$
$$(A_{04} = e \wedge A_{23} = a \wedge A_{33} = e) \vee (A_{04} = h \wedge A_{31} = i \wedge A_{33} = e) \vee (A_{12} = h \wedge A_{33} = i) \Rightarrow C_8$$
$$(A_{15} = c \wedge A_{23} = f \wedge A_{33} = g) \Rightarrow C_9$$
$$(A_{12} = c \wedge A_{33} = i) \vee (A_{23} = a \wedge A_{33} = g) \vee (A_{21} = i \wedge A_{23} = h \wedge A_{33} = j) \Rightarrow C_{10}$$

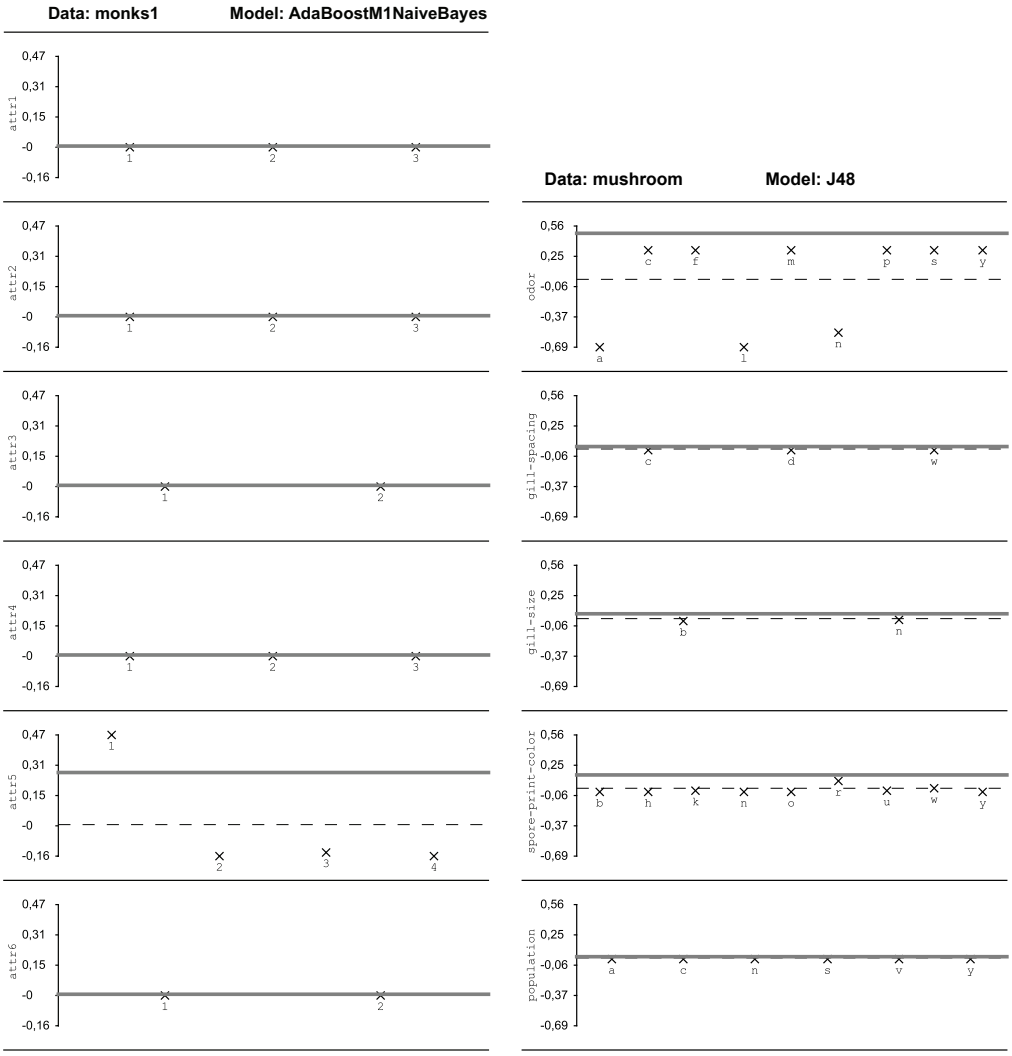


Figure 8: Model visualizations for the Naive Bayes model trained on Monks1 (right-hand side) and a decision tree on the mushroom data set (left-hand side).